Gnutella Style Peer-to-Peer (P2P) File Sharing

A JAVA Application (Test File)

by

Abhishek Kumar

Table of Contents

DISCI

The project has been tested under HPCC cluster environment and all peers are running successfully. The image below shows the cloning of the project form my local machine to the HPCC cluster

```
Abhishek@DESKTOP-C2AUL59 MINGW64 ~/Desktop/Computer Science Texas Tech Universit
y/Spring 2018/CS5352 Advanced Operating Systems/Project3_Gnutella
$ scp -r Gnutella/ abhiskum@disci.hpcc.ttu.edu:
abhiskum@disci.hpcc.ttu.edu's password:
compiler.xml                                  100%  529    88.3KB/s   00:00
misc.xml                                      100%  513    85.7KB/s   00:00
workspace.xml                                 100%   17KB   1.6MB/s   00:00
config.properties                             100%  499    62.3KB/s   00:00
Gnutella.iml                                  100%   80    13.3KB/s   00:00
mesh.properties                               100%  981   161.7KB/s   00:00
```

```
[abhiskum@disci ~]$ ls
ansible-swarm-baremetal-single-node  hello     project1
Gnutella                             project0  simple-shell.c
[abhiskum@disci ~]$
```

1. GitHub Link

Given below is the GitHub link of the Gnutella style peer-to-peer (P2P) file sharing system source file, design pdf and test pdf.

https://github.com/abhishek-kumar-code/Gnutella_Style_Peer-to-Peer-P2P-_FileSharing/tree/master/Gnutella

https://github.com/abhishek-kumar-code/Gnutella_Style_Peer-to-Peer-P2P-_FileSharing/blob/master/Gnutella/design.pdf

https://github.com/abhishek-kumar-code/Gnutella_Style_Peer-to-Peer-P2P-_FileSharing/blob/master/Gnutella/test.pdf

2. Maven Build - Building the project Gnutella using Star Topology

The following document explains how to build the project and execute the test cases.

The build tool used is **Maven**.
Instructions to build and run the Gnutella style peer-to-peer (P2P) file sharing system is discussed below.

Main

- Open command prompt and go to Gnutella folder that contains pom (XML document). The path file for my machine is given below:
  C:\Users\Abhishek\Desktop\Computer Science Texas Tech University\Spring 2018\CS5352 Advanced Operating Systems\Project3_Gnutella\Gnutella
- In the command prompt we type the command **mvn clean**. The Maven Clean Plugin, as the name implies, attempts to clean the files and directories generated by Maven during its build. This is shown in the figure below.

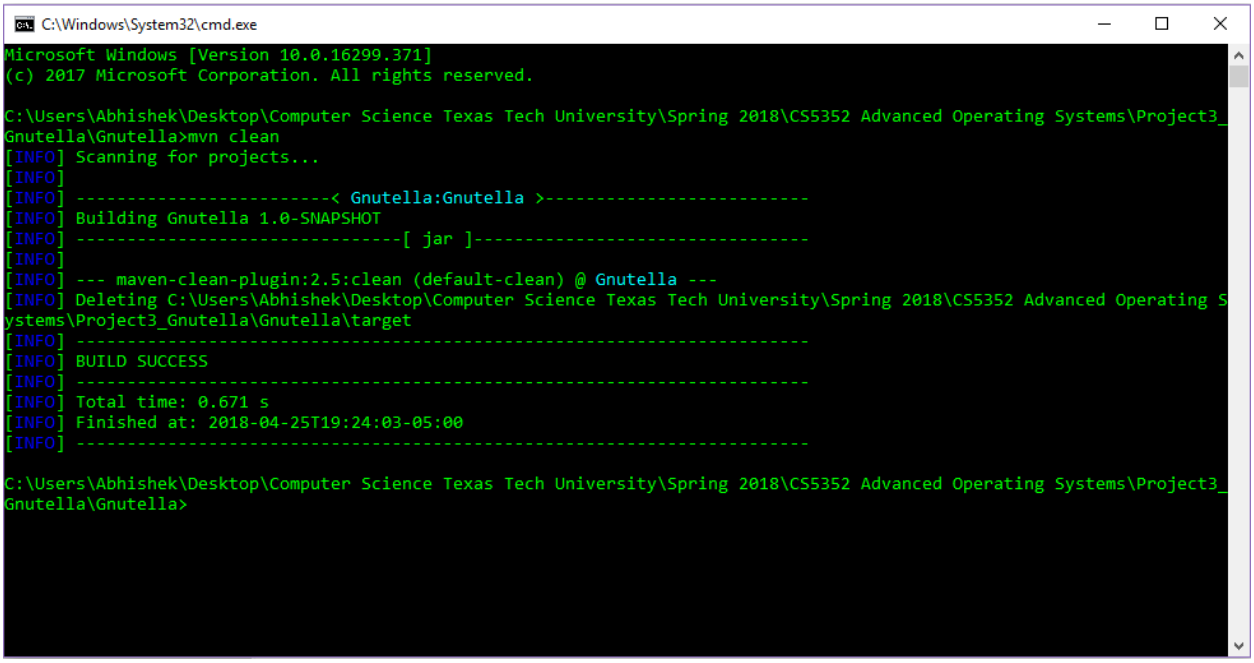

Fig 1: Output from the command mvn clean

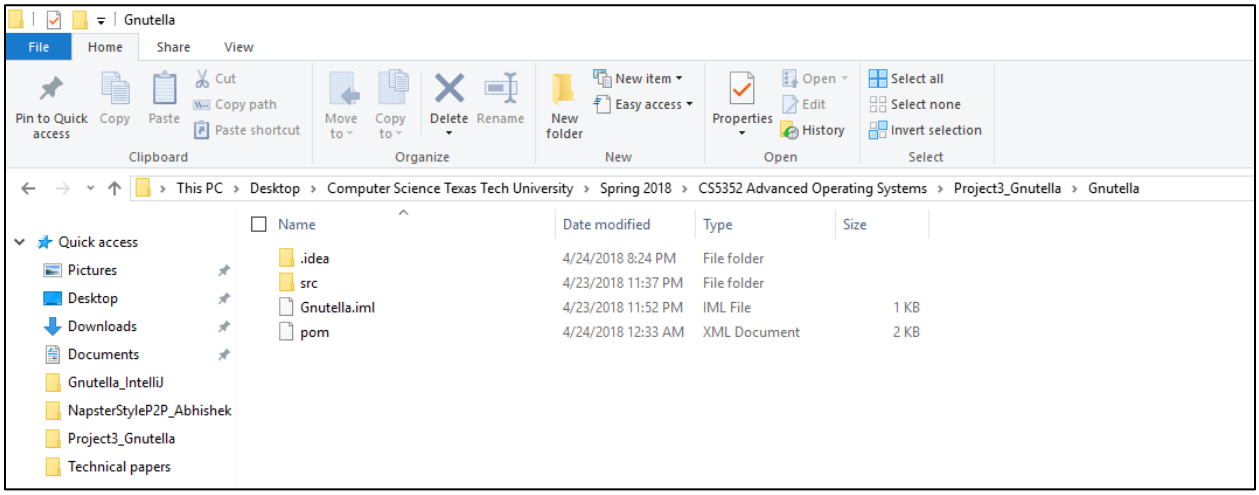

Fig 2: Gnutella folder after running mvn clean

- Now, in the command prompt we type the command **mvn install**. This command installs the package into the local repository, for use as a dependency in other projects locally. We can clearly see that the Gnutella folder now has a target folder which contains the jar file. This format is common for distributing programs and libraries that are written in Java. Our .java files is compiled by the JVM to .class files.

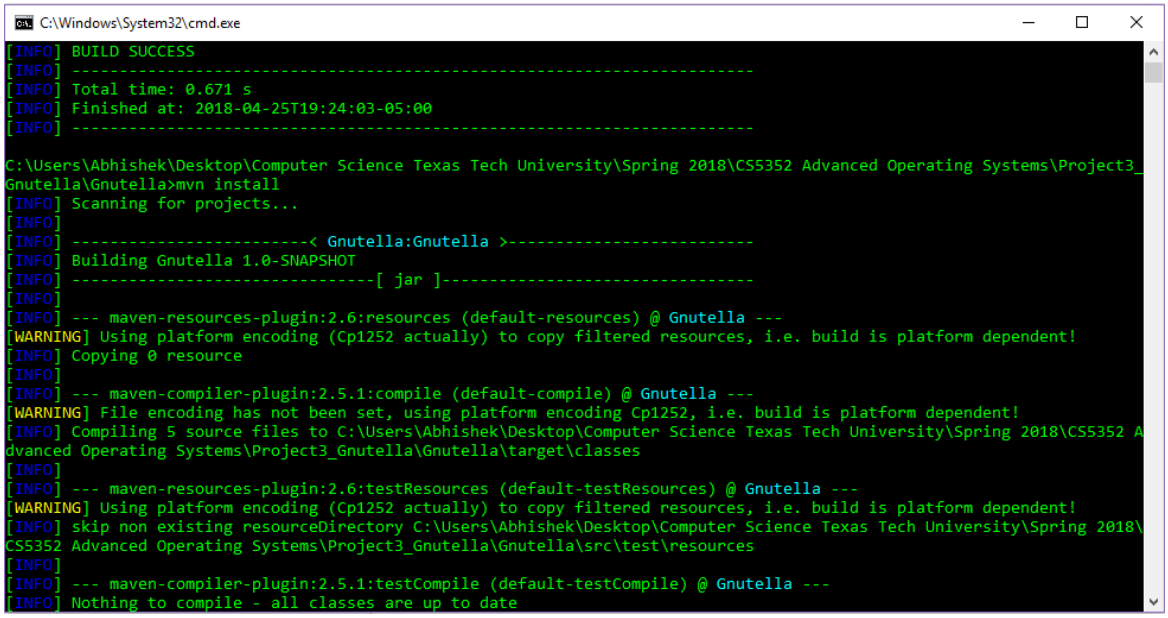

Fig 3: Output from the command mvn install

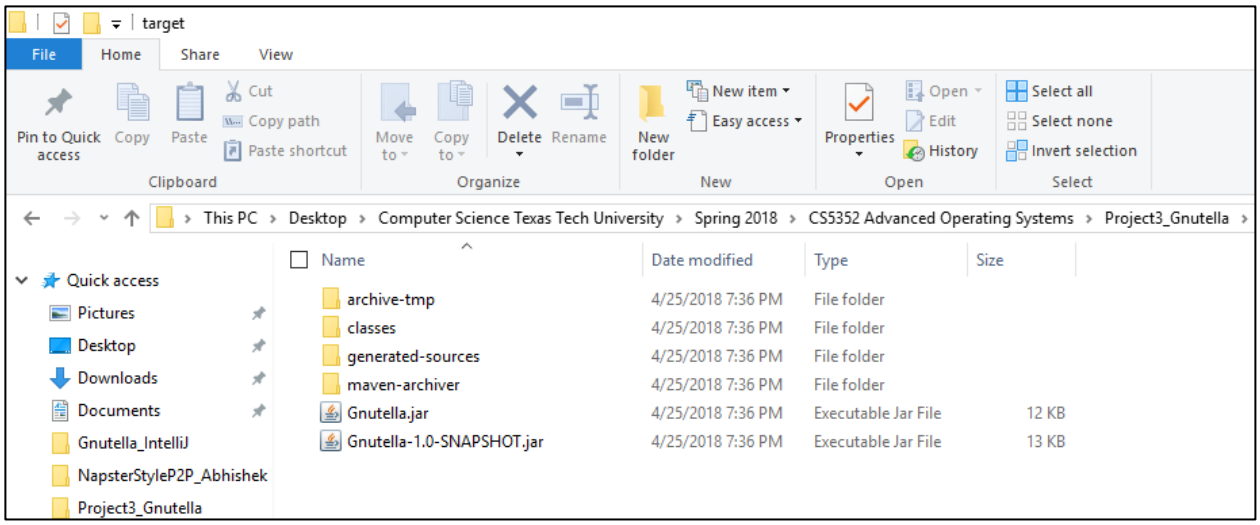

Fig 4: Target folder after running mvn clean

- **IMPORTANT**: Copy the **config.properties** file and the **mesh.properties** file from the Gnutella folder into the newly created target folder that contains the executable jar file.
- Open a new command prompt and go to target folder (under Gnutella folder) that contains the executable jar file. The path file for my machine is given below: *C:\Users\Abhishek\Desktop\Computer Science Texas Tech University\Spring 2018\CS5352 Advanced Operating Systems\Project3_Gnutella\Gnutella\target*
- In the command prompt we type the command **java –jar Gnutella.jar**. This runs the project as shown below. Follow the instructions as shown in the Fig 6 to run the project.

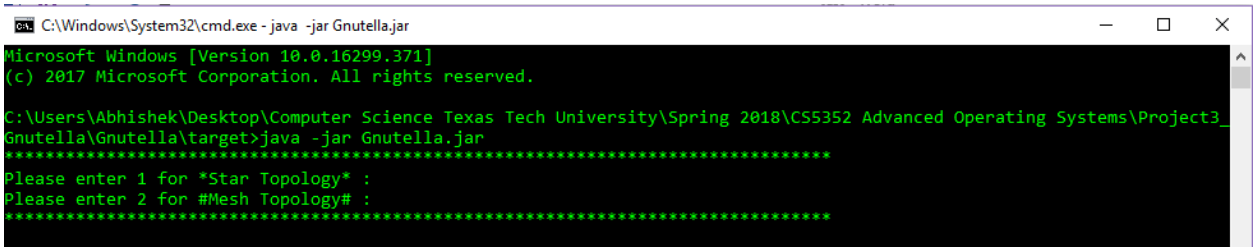

Fig 5: Output from the command java –jar Gnuetella.jar

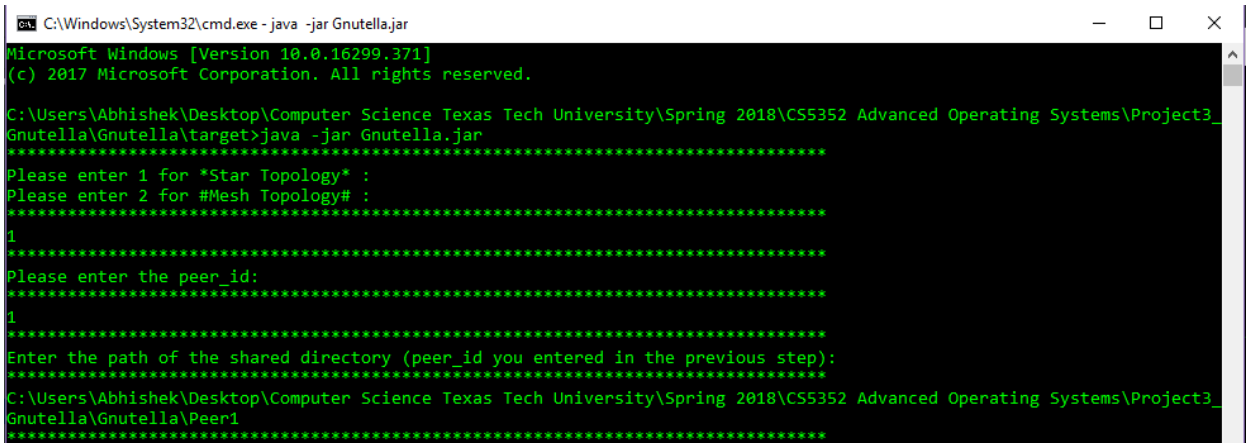

Fig 6: Star Topology distributed system file sharing (connecting peer#1)

- Now open a new command prompt and go to target folder (under Gnutella folder) that contains the executable jar file. The path file for my machine is given below: *C:\Users\Abhishek\Desktop\Computer Science Texas Tech University\Spring 2018\CS5352 Advanced Operating Systems\Project3_Gnutella\Gnutella\target*
- In this new command prompt we type the command **java –jar Gnutella.jar**. This runs the project as shown below. Here we run the same topology as show in Fig 6 with a different peer_id (say peer#2)

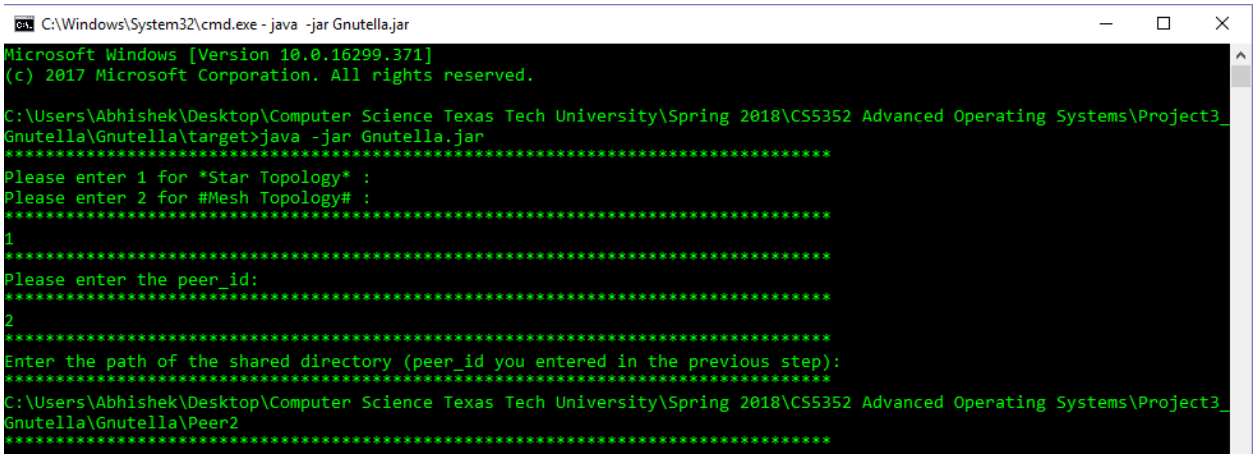

Fig 7: Star Topology distributed system file sharing (connecting peer#2)

- Now go back to the previous command prompt (as shown in Fig 6) and enter the file that needs to be searched and downloaded to peer#1.

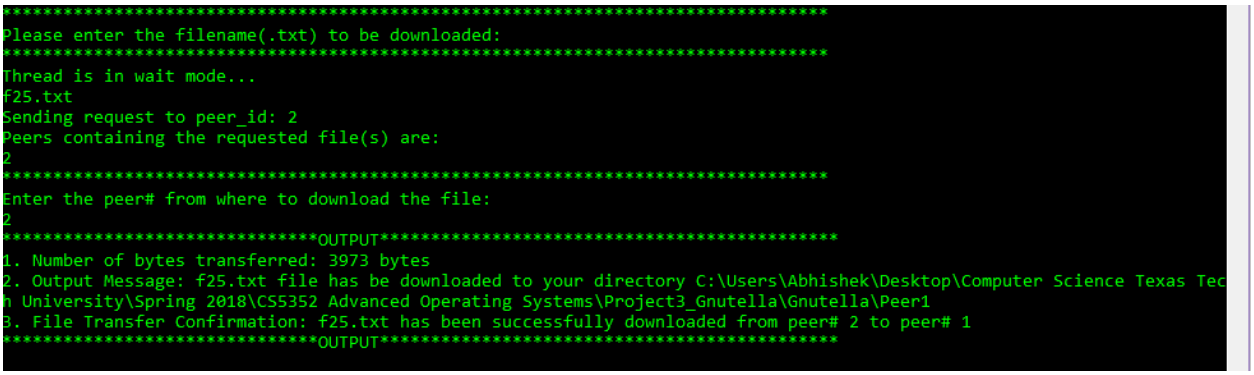

Fig 8: Star Topology P2P file transfer in distributed system (command prompt with peer#1)

As shown in Fig 9, the file of size 3973 bytes is transferred to the folder Peer1.

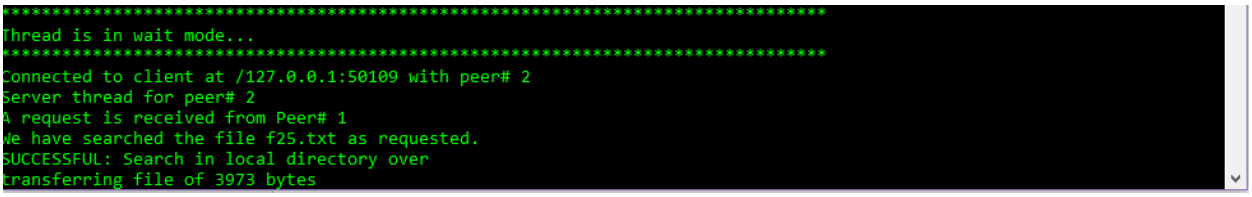

Fig 9: Star Topology P2P file transfer in distributed system (command prompt with peer#2)

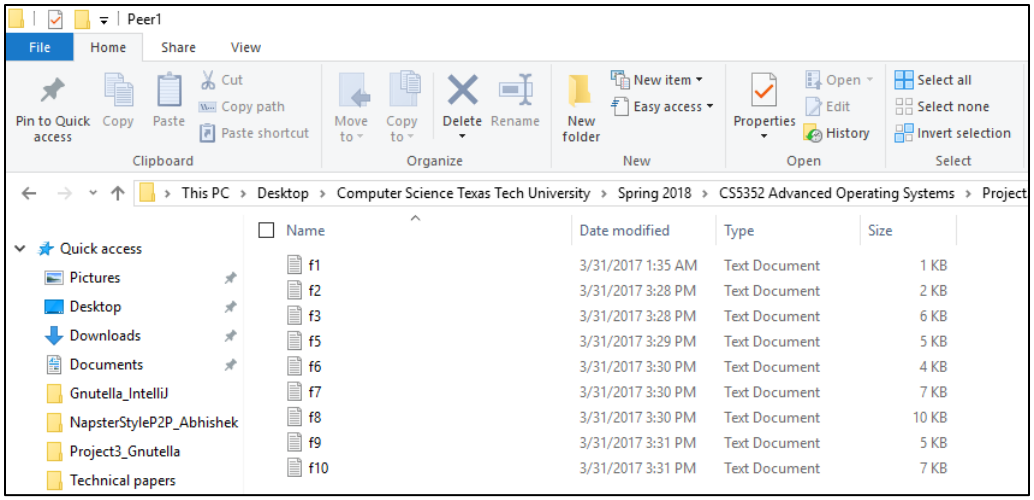

Fig 10: Peer 1 folder before file search and download

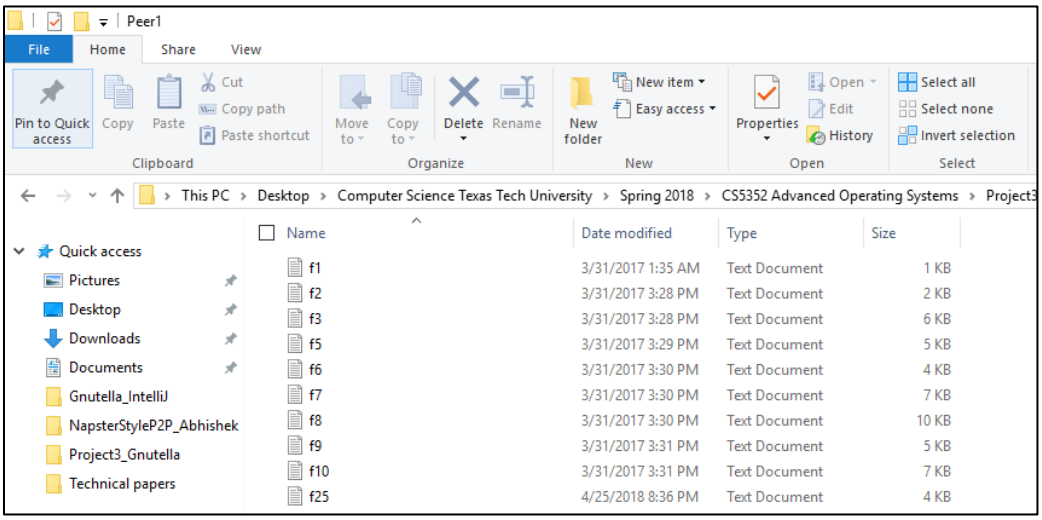

Fig 11: Peer 2 folder before file search and download

3. Maven Build - Building the project Gnutella using Mesh Topology

The following execution was performed on star topology. Using the similar instructions we can select option#2 if we want to perform mesh topology. The mesh topology will parse the mesh.properties file instead of config.properties file. This can be seen in the main.java code where it enters the else condition when choice given by user is 2. You can see the details of the mesh.properties file in the design documentation section (section 3, page#6, 7). The figure below depicts which peers communicate among each other in our distributed system.

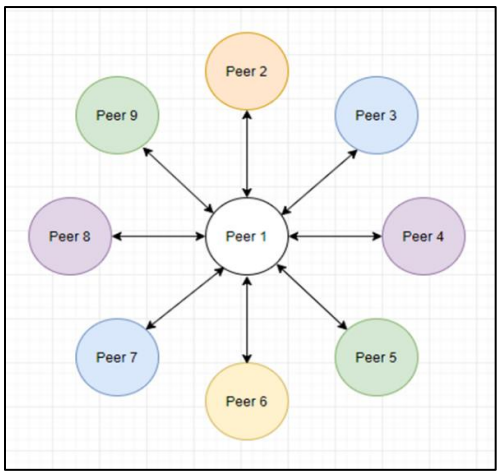

Fig 12: Star Topology

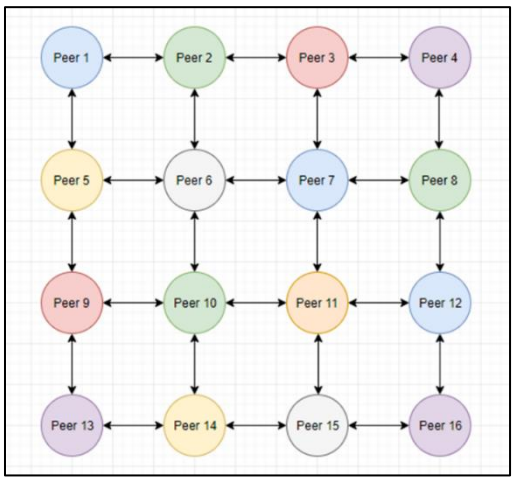

Fig 13: Mesh Topology

4. Performance Test

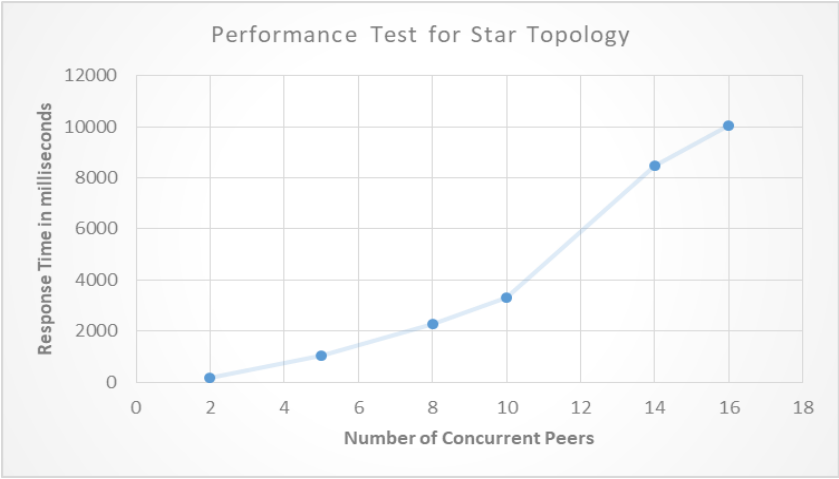

Fig 14 Performance Test for Star Topology